High Integrity Computing (F23PS2)
High Integrity Software Development (F29HD2)

A Note on Flow Analysis for Arrays within the SPARK Examiner

Andrew Ireland
Department of Computer Science
School of Mathematical & Computer Sciences
Heriot-Watt University

For the purposes of flow analysis, the SPARK Examiner treats an array as a single entity. Some of the consequences of this aspect of flow analysis are explored in this note.

1 A simple example

Consider the following package specification and body. Note in particular the definition and usage of the array `Data`:

```
package ArrayInitA
   --# own Data;
   --# initializes Data;
is
   Data_Size: constant:= 2;
   subtype Index_Range is Integer range 1..Data_Size;

   procedure Data_Update(X: in Integer; Y: in Index_Range);
   --# global out Data;
   --# derives Data from X, Y ;

end ArrayInitA;

package body ArrayInitA
is
   type Vector is array(Index_Range) of Integer;
   Data: Vector;

   procedure Data_Update(X: in Integer; Y: in Index_Range)
   is
   begin
      Data(Y):=X;
   end Data_Update;

begin
   Data(1):= 0;
   Data(2):= 0;
end ArrayInitA;
```

The SPARK Examiner detects flow analysis errors with respect to the package body given above. The listing file generated for the the package body is given below:

```
          *******************************************************
                           Listing of SPARK Text
       SPARK95 Examiner with VC and RTC Generator Release 6.0 / 11.01
                           Demonstration Version
          *******************************************************


                       DATE : 11-MAY-2005 11:15:13.32

Line
   1
   2  package body ArrayInitA
   3  is
   4     type Vector is array(Index_Range) of Integer;
   5     Data: Vector;
   6
   7     procedure Data_Update(X: in Integer; Y: in Index_Range)
   8     is
   9     begin
  10        Data(Y):=X;
            ^1
!!! (  1)  Flow Error        : 23: Statement contains reference(s) to undefined
           variable Data.

  11     end Data_Update;

??? (  2)  Warning           :602: The undefined initial value of Data may be used
           in the derivation of Data.

  12
  13  begin
  14     Data(1):= 0;
         ^3
!!! (  3)  Flow Error        : 23: Statement contains reference(s) to undefined
           variable Data.

  15     Data(2):= 0;
  16  end ArrayInitA;

??? (  4)  Warning           :602: The undefined initial value of Data may be used
           in the derivation of Data.

  17
  18


--End of file--------------------------------------------------
```

Note that the warnings include a reference to the package initialization code, *i.e.* flow analysis claims that the array `Data` is undefined. Because flow analysis treats an array as a single entity, it requires all elements to be initialized simultaneously, *e.g.* when `Data(1)` is assigned the value `0`, the Examiner spots that `Data(2)` is undefined.

2 First revised example

But how can we initialize all the elements of `Data` simultaneously? Simultaneous assignment is achieved using the aggregate construct, *e.g.*

```
Data:= Vector'(Index_Range => 0);
```

This has the effect of assigning the value of `0` to every element of `Data` simultaneously. The revised code is given below:

```
package ArrayInitB
   --# own Data;
   --# initializes Data;
is
   Data_Size: constant:= 2;
   subtype Index_Range is Integer range 1..Data_Size;

   procedure Update_Data(X: in Integer; Y: in Index_Range);
   --# global out Data;
   --# derives Data from X, Y ;

end ArrayInitB;

package body ArrayInitB
is
   type Vector is array(Index_Range) of Integer;
   Data: Vector;

   procedure Update_Data(X: in Integer; Y: in Index_Range)
   is
   begin
      Data(Y):=X;
   end Update_Data;

begin
   Data:= Vector'(Index_Range => 0);
end ArrayInitB;
```

But the SPARK Examiner still finds an error, as shown below:

```
          *******************************************************
                            Listing of SPARK Text
       SPARK95 Examiner with VC and RTC Generator Release 6.0 / 11.01
                            Demonstration Version
          *******************************************************


                        DATE : 11-MAY-2005 11:15:48.70

Line

   package body ArrayInitB
   is
      type Vector is array(Index_Range) of Integer;
      Data: Vector;

      procedure Update_Data(X: in Integer; Y: in Index_Range)
      is
      begin
```

```
           Data(Y):=X;
           ^1
!!! (  1)  Flow Error        : 23: Statement contains reference(s) to undefined
           variable Data.

        end Update_Data;

??? (  2)  Warning           :602: The undefined initial value of Data may be used
           in the derivation of Data.


     begin
        Data:= Vector'(Index_Range => 0);
     end ArrayInitB;

+++        Flow analysis of package initialization
           performed: no errors found.




--End of file--------------------------------------------------
```

The remaining error relates to the array assignment within the body of the procedure `Update_Data`. The problem is again a consequence of how flow analysis treats arrays. Note that when the $\texttt{Y}^{th}$ element of `Data` is updated, *i.e.* `Data(Y):= X`, all the other elements remain unchanged. From the perspective of flow analysis, the initial values of the unchanged elements correspond to their final values. As a consequence, flow analysis concludes that the final value of `Data` is partially derived from the initial value of `Data`. However, this is **not** reflected in the `global` and `derives` annotations associated with `Update_Data` – see the package specification for `ArrayInitB` given above.

3 Second revised example

As noted, the annotations for `Update_Data` must reflect the fact that the derivation of `Data` will use initial values from `Data`. This involves annotating `Data` as an input, as shown below in our revised package specification:

```
package ArrayInitC
   --# own Data;
   --# initializes Data;
is
   Data_Size: constant:= 2;
   subtype Index_Range is Integer range 1..Data_Size;

   procedure Update_Data(X: in Integer; Y: in Index_Range);
   --# global in out Data;
   --# derives Data from Data, X, Y ;

end ArrayInitC;

package body ArrayInitC
is
   type Vector is array(Index_Range) of Integer;
   Data: Vector;

   procedure Update_Data(X: in Integer; Y: in Index_Range)
```

```
   is
   begin
      Data(Y):=X;
   end Update_Data;

begin
   Data:= Vector'(Index_Range => 0);
end ArrayInitC;
```

The subsequent analysis by the SPARK Examiner succeeds, as shown below.

```
          *******************************************************
                          Listing of SPARK Text
       SPARK95 Examiner with VC and RTC Generator Release 6.0 / 11.01
                          Demonstration Version
          *******************************************************


                      DATE : 11-MAY-2005 11:16:19.36

Line
      package body ArrayInitC
      is
         type Vector is array(Index_Range) of Integer;
         Data: Vector;

         procedure Update_Data(X: in Integer; Y: in Index_Range)
         is
         begin
            Data(Y):=X;
         end Update_Data;

+++        Flow analysis of subprogram Update_Data
           performed: no errors found.

      begin
         Data:= Vector'(Index_Range => 0);
      end ArrayInitC;

+++        Flow analysis of package initialization
           performed: no errors found.


--End of file--------------------------------------------------
```